robotron

Programmtechnische
Beschreibung

Anleitung für den Bediener

Programmiersystem
Standard-BAS I C

VEB Robotron-Projekt Dresden

C 3015-0300-1 M 3030

Arbeitsplatzcomputer A 7150
Personalcomputer EC 1834

Betriebssystem DCP 1700
Betriebssystem DCP 3.20

Die vorliegende Anwenderdokumentation „Anleitung für den Bediener" Standard-BASIC entspricht dem Stand von 11/87.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Herausgeber:
VEB Robotron-Projekt Dresden
Leningrader Straße 9
Dresden
8010

© 1987 robotron

Kurzreferat

In der vorliegenden „Anleitung für den Bediener" wird die Bedienung des Programmiersystems für Standard-BASIC auf dem Arbeitsplatzcomputer A7150 **und** auf dem Personalcomputer EC1834 unter dem Betriebssystem DCP beschrieben.

Kenntnisse der Sprache Standard-BASIC, die in der Anwenderdokumentation C3016-0300-1 M3030 „Standard-BASIC Sprachbeschreibung" definiert sind, und elementare Kenntnisse des Betriebssystems DCP werden vorausgesetzt.

Die Schrift enthält eine Übersicht über den Aufbau des Systems und beschreibt detailliert die zur Verfügung stehenden Kommandos. Weiterhin wird erläutert, wie eine Programmverkettung zu realisieren ist und welche Fehlermeldungen bei der Arbeit **mit** dem System auftreten können.

Inhaltsverzeichnis

		Seite
1.	Aufbau des Standard-BASIC-Programmiersystems für DCP	4
2.	Start, Restart und Verlassen des Programmiersystems	6
3.	Kommandos	8
3.1.	Eingabe von Programmen über das-Bedienterminal	8
3.2.	Zeilenbereiche und Dateispezifikationen	9
3.3.	Ausgabe des Programmtextes auf das Bedienterminal (LIST)	10
3.4.	Ausgabe der Programmliste (PROTOCOL)	11
3.5.	Ausgabe der Fehlerliste (ERRORLIST)	12
3.6.	Eingabe eines Programms von einer Datei (OLD)	12
3.7.	Hinzufügen von Programmzeilen aus einer Datei (ADD)	13
3.8.	Ausgabe des aktuellen Programms in eine Datei (SAVE)	14
3.9.	Auswahl des aktuellen Gerätes (LOG)	15
3.10.	Löschen von Zeilen (DELETE)	15
3.11.	Extrahieren von Zeilen (EXTRACT)	**16**
3.12.	Neunummerieren von Zeilen (RENUMBER)	17
3.13.	Laden und Abarbeiten des aktuellen Programms (RUN)	18
3.14.	Wiederholte Abarbeitung des aktuellen Programms (START)	**18**
3.15.	Übersetzung des Programms (TRANSLATE)	18
4.	Programmverkettung mit der CHAIN-Anweisung	19
5.	Systemmitteilungen	20
5.1.	Mitteilungen bei der Programmerstellung	20
5.2.	Mitteilungen über Fehler bei der Arbeit mit dem BASIC-Arbeitsspeicher	22
5.3.	Mitteilungen bei der Programmübersetzung	22
5.3.1.	Mitteilungen bei der lexikalen Analyse	22
5.3.2.	Syntaktische Fehler	23
5.3.3.	Semantische Fehler	25
5.4.	Mitteilungen bei der Programmabarbeitung	30
6.	Unterbrechung des Programmlaufs	32

Bildverzeichnis

Bild 1: Bestandteile des BASIC-Programmiersystems 5
Bild 2: Speicheraufteilung bei der Arbeit mit dem BASIC-Programmiersystem 5

1. Aufbau des Standard-BASIC-Programmiersystems für DCP

Zur sicheren Beherrschung des BASIC-Systems sind einige Kenntnisse über den Aufbau und die Arbeitsweise des Systems erforderlich. Diese Kenntnisse werden in dem folgenden Abschnitt vermittelt.

Das BASIC-Programmiersystem besteht aus folgenden Komponenten:

Kommandoprogramm BAS.EXE

Das Kommandoprogramm BAS.EXE führt die Kommunikation mit dem Benutzer, wertet die eingegebenen Kommandos aus und aktiviert nach. Bedarf die übrigen Komponenten des Programmiersystems.

Editor BED.OVR

Die Editorkomponente enthält Mittel zur Eingabe, Sortierung, Verwaltung und Ausgabe von BASIC-Programmen. Die entsprechenden Kommandos sind im Abschnitt 3. beschrieben.

Übersetzerbestandteile BTR.OVR und BP2.OVR

Mit dem Übersetzer werden die BASIC-Quellprogramme in einen ausführbaren Zwischencode (rpn-Code) überführt. BTR.OVR kontrolliert das BASIC-Programm auf syntaktische und semantische Fehler, bevor BP2.OVR das Zwischenprogramm herstellt. Die Übersetzung der BASIC-Programme vor ihrer Ausführung ist notwendig, da wegen des gewachsenen Sprachumfangs von Standard-BASIC eine direkte Abarbeitung des Programmtextes durch einen Textinterpreter nicht mehr mit vertretbarem Aufwand möglich ist.
Die Übersetzung erfolgt immer dann, wenn das Programm abgearbeitet werden soll und kein aktuelles Zwischenprogramm vorliegt oder wenn die Übersetzung durch Kommando angewiesen wird. Ein vorhandenes Zwischenprogramm geht verloren, wenn der zugehörige Programmtext verändert wird.

Zwischenprogramminterpreter BRU.OVR

Der Zwischenprogramminterpreter führt das vom Übersetzer erzeugte Zwischenprogramm aus. Er enthält alle benötigten Routinen für die Ein- und Ausgabe von Daten und für die in den Programmen enthaltenen Berechnungen.

Der Datenaustausch zwischen den einzelnen Bestandteilen des BASIC-Programmiersystems erfolgt über den BASIC-Arbeitsspeicher.
Der BASIC-Arbeitsspeicher enthält u.a. den Text des aktuellen BASIC-Programms, das dazu gehörende ausführbare Zwischenprogramm und die Übersetzertabellen. Der BASIC-Arbeitsspeicher wird beim BASIC-Programmiersystem für DCP im Hauptspeicher geführt.
Während der Arbeit mit dem BASIC-Programmiersystem bleiben nur das Kommandoprogramm und der BASIC-Arbeitsspeicher ständig im Hauptspeicher. Die OVR-Teile überlagern einander und werden bei Bedarf in den Hauptspeicher geladen. Für eine zügige Arbeit mit dem BASIC-Programmiersystem sollten die OVR-Teile so untergebracht werden, daß sie möglichst schnell nachgeladen werden können. Am günstigsten ist dazu eine virtuelle Magnetplatte. Hierbei ist jedoch zu beachten, daß die virtuelle Magnetplatte den Platz begrenzt, der als BASIC-Arbeitsspeicher verwendet werden kann (siehe Bild 2).

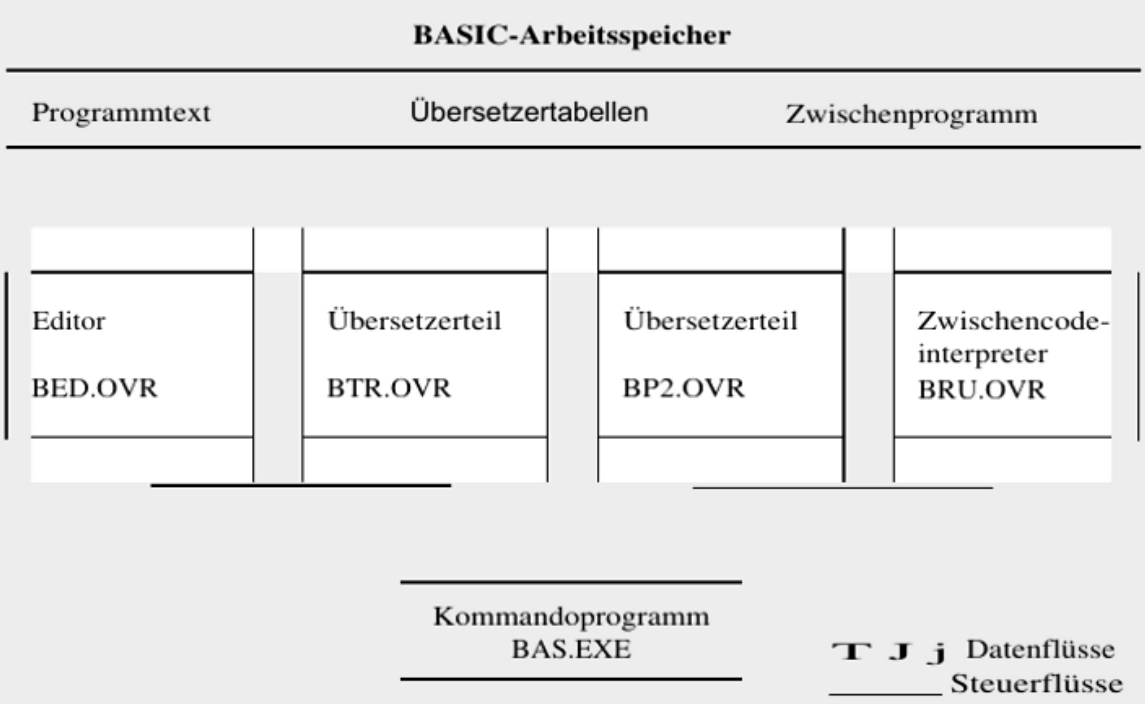

Bild 1: Bestandteile des BASIC-Programmiersystems

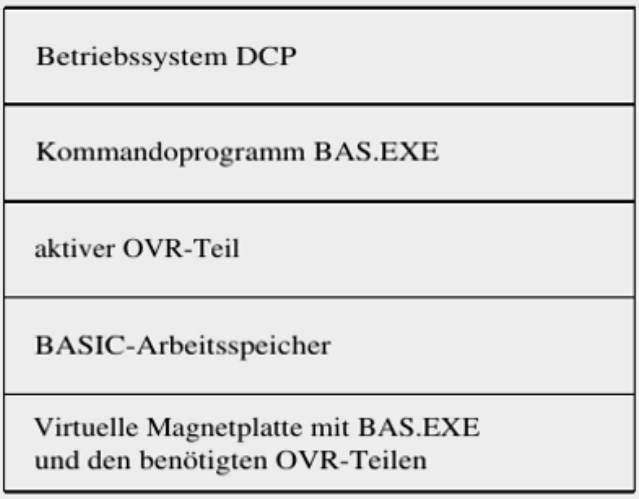

Bild 2: Speicheraufteilung bei der Arbeit mit dem BASIC-Programmiersystem

Muß auf die virtuelle Magnetplatte verzichtet werden, sollten BAS.EXE und OVR-Teile auf der Festplatte untergebracht werden. Das Nachladen der OVR-Teile erfolgt zunächst von dem Gerät, von dem aus BAS.EXE gestartet wurde. Deshalb sollte das BASIC-Programmiersystem mit allen seinen Bestandteilen auf einem Gerät untergebracht werden. Wird ein nachzuladender OVR-Teil auf diesem Gerät nicht gefunden, wird mit der Ausschritt

ENTER DEVICE AND PATH FOR PROCESSOR XXX.OVR

ein neues Gerät und/oder eine neue Pfadangabe angefordert. Das Nachladen wird mit dem angegebenen Gerät bzw. Pfad wiederholt. Wird anstelle eines Gerätenamens mit (CR) quittiert, wird das Nachladen abgebrochen und es erscheint nach der Ausschrift

*** PROCESSOR XXX.OVR NOT AVAILABLE

wieder die Kommandoanforderung „—".

2. Start, Restart und Verlassen des Programmiersystems

Das BASIC-Programmiersystem wird gestartet, indem der Kommandoprozessor aktiviert wird, z. B. durch:

BAS – Start vom aktuellen Gerät
E: BAS – Start von E:

Nach dem Start erscheint die Anschrift:

Robotron Standard-BASIC, Version XX.XX DCP, 1988
© Copyright VEB Robotron-Projekt Dresden

und die Eingabeanforderung:

INDENTIFICATION

Nach dieser Eingabeanforderung ist der Identifikationscode einzugeben. Der Identifikationscode ist ein Identifikator, der mit einem Buchstaben beginnen muß. Es können beliebig viele Buchstaben oder Ziffern folgen, jedoch werden nur die ersten sechs Zeichen berücksichtigt. Kleinbuchstaben werden in Großbuchstaben umgewandelt. Das Unterstreichungszeichen ist nicht zulässig.

Der Identifikationscode wird wie folgt verwendet:

- Er ersetzt im Hauptprogramm, in dem die PROGRAM-Anweisung fehlt, den Namen der Programmeinheit (siehe Sachbeschreibung Abschnitt 4.3.). In diesem Fall darf mit dem Identifikationscode in dem betreffenden Hauptprogramm kein anderes Objekt bezeichnet werden.
- Er wird im SAVE- und im OLD-Kommando (s. Abschn. 3.2.) anstelle des Dateinamens verwendet, wenn dort kein Dateiname angegeben wurde.
- Er wird als Dateiname verwendet, wenn der BASIC-Arbeitsspeicher bei einem BYE- oder RESTART-Kommando als Datei gesichert wird.
- Er dient zur Kennzeichnung von Programmlistings beim PROTOCOL-Kommando (s. Abschn. 3.2.).

  Es ist zweckmäßig, als Bezeichnung für das Hauptprogramm in der PROGRAM-Anweisung für den Dateinamen des Quellprogramms und den Identifikationscode ein und denselben Identifikator zu wählen.

Nach Eingabe des Identifikationscodes erscheint die Kommandoanforderung

Das BASIC-Programmiersystem ist damit betriebsbereit und erwartet die Eingabe von BASIC-Zeilen und -Kommandos.

Soll die Arbeit mit dem BASIC-Programmiersystem beendet werden, gibt man das BYE-Kommando ein.

Syntax

bye-kommando:: = BYE (delete-schalter)

delete-schalter:: = /DE
/-DE

Die Unterstreichung des Buchstabens B im Kommando bedeutet, daß das Kommandoschlüsselwort BYE bis auf B abgekürzt werden kann. Möglich sind also B, BY oder BYE.

Kleinbuchstaben sind im Kommandoschlüsselwort zulässig und werden zu Großbuchstaben umgewandelt.

Die Unterstreichung wird auch im Abschnitt 3. zur Kennzeichnung von Abkürzungsmöglichkeiten verwendet.

Der Delete-Schalter im BYE-Kommando steuert das Sichern des BASIC-Arbeitsspeichers bei Beendigung der Arbeit mit dem Programmiersystem. Wird /-DE angegeben oder fehlt der Delete-Schalter, wird der BASIC-Arbeitsspeicher vor dem Verlassen des Systems in eine Datei gesichert. Die Datei wird auf dem aktuellen Gerät angelegt. Sie hat den Typ .DAT. Als Dateiname wird der Identifikationscode verwendet. Auf den Inhalt des gesicherten Arbeitsspeichers kann man sich beim Start des BASIC-Systems beziehen.

Existiert beim Start zu einem eingegebenen Identifikationscode auf dem aktuellen Gerät eine Datei mit dem entsprechenden Namen, wird der BASIC-Arbeitsspeicher aus dieser Datei wiederhergestellt. Vor der Ausgabe der ersten Kommandoforderung erscheint in diesem Fall noch die Ausschrift:

READ VS-FILE identifikationscode. DAT

Die Arbeit kann dann an dem Punkt fortgesetzt werden, bei dem sie mit dem BYE/-DE-Kommando abgeschlossen wurde, d. h., ein vorhandenes abarbeitbares Zwischenprogramm, Informationen zum Programmtext u. a, bleiben bei BYE/-DE erhalten.

Wird der Schalter /DE angegeben, so wird der BASIC-Arbeitsspeicher nicht gesichert. Eine eventuell auf dem aktuellen Gerät vorhandene Kopie des Arbeitsspeichers wird gelöscht.

Die Standardannahme /-DE bei BYE gewährleistet, daß der Text des aktuellen Programms auch dann nicht verlorengeht, wenn das Abspeichern des geänderten oder neu erstellten Programms mit dem SAVE-Kommando (s. Abschn. 3.) vergessen wurde.

Es sei jedoch darauf verwiesen, daß der gesicherte BASIC-Arbeitsspeicher wesentlich mehr Platz auf der Magnetplatte belegt, als der Programmtext.

Weiterhin ist zu beachten, daß ein gesicherter BASIC-Arbeitsspeicher nur durch dieselbe Version des BASIC-Programmiersystems wieder eingelesen werden kann.

Beim Restart-Kommando sind die Funktionen für Beendigung und Start des Systems vereinigt.

Syntax

restart-kommando:: = RESTART [delete-schalter]

delete-schalter:: = /DE
/-DE

Ist der Schalter /-DE oder kein Schalter angegeben, wird wie beim BYE-Kommando der BASIC-Arbeitsspeicher in eine Datei gerettet. Ist /DE angegeben, geht sein Inhalt verloren und eine eventuell auf dem aktuellen Gerät vorhandene Kopie des Arbeitsspeichers wird gelöscht.

Anschließend wird mit

IDENTIFICATION:

ein neuer Identifikationscode angefordert. Falls unter diesem Identifikationscode ein auf Datei gesicherter BASIC-Arbeitsspeicher vorliegt, wird dieser gelesen.

Die Sicherung des BASIC-Arbeitsspeichers als Datei erfolgt vor allem dann, wenn:

- ein fertiges BASIC-Programm häufig abgearbeitet werden soll,
- die Arbeit an einem Problem kurzfristig beendet werden muß und später an gleicher Stelle wieder aufgenommen werden soll,
- Programme übersetzt wurden, die von anderen Programmen mit der CHAIN-Anweisung gestartet werden sollen. (Der Programmbeschreiber in der CHAIN-Anweisung bezieht sich stets auf den gesicherten Arbeitsspeicher mit dem zu startenden Programm.)

3. Kommandos

Der folgende Abschnitt beschreibt die zur Bedienung des BASIC-Programmiersystems verfügbaren Kommandos. Darunter befinden sich alle Kommandos des Editormoduls des Standards ANSI X3. 113-1987.

Das sind das LIST-, DELETE-, EXTRACT- und RENUMBER-Kommando, die vor allem bei der Erstellung und Modifizierung von BASIC-Programmen verwendet werden. Weitere Kommandos dienen zur Ein- und Ausgabe von Programmtexten auf Magnetplatten, zur Listenausgabe und zur Programmabarbeitung.

3.1. Eingabe von Programmen über das Bedienterminal

Nach der Kommandoanforderung „--" können BASIC-Zeilen eingegeben werden. Sie werden entsprechend ihrer Zeilennummer durch den Editor des Programmiersystems sortiert, so daß die Eingabefolgen

```
30 END
10 PROGRAMM druck
20 PRINT „Guten Tag"
```

und

```
10 PROGRAM druck
20 PRINT „Guten Tag"
30 END
```

den gleichen Programmtext ergeben. Ist eine Zeile mit einer bestimmten Zeilennummer bereits vorhanden und es wird erneut eine Zeile mit dieser Zeilennummer eingegeben, so überschreibt die eingegebene Zeile die vorhandene Zeile. Eingegebene Fortsetzungszeilen werden der unmittelbar vorher eingegebenen BASIC-Zeile zugeordnet. Dabei wird erst bei der Übersetzung des Programms überprüft, ob diese Zeile mit einem Ampersand „&" endet. Zu Beginn der Arbeit mit dem Programmiersystem oder nach Kommandos werden Fortsetzungszeilen nicht akzeptiert.

Liegt eine Zeilennummer nicht im Bereich von 1 bis 50000 oder hat eine Zeilennummer mehr als fünf Ziffern, erscheint die Meldung

```
* * * ILLEGAL LINE NUMBER
      ENTER THE CORRECT LINE NUMBER
      (5 DIGITS AT MOST; >0; <=50000)
```

und es wird mit „:" eine korrekte Zeilennummer angefordert. Nach Eingabe einer Zeilennummer wird die Zeile unter dieser Zeilennummer eingefügt. Anstelle einer Zeilennummer kann auch <CR, eingegeben werden. Dann wird die beanstandete Zeile nicht eingefügt.

3.2. Zeilenbereiche und Dateispezifikationen

Der Bezug auf einzelne Programmzeilen im BASIC-Arbeitsspeicher erfolgt durch die Angabe ihrer Zeilennummer, der Bezug auf Programmabschnitte durch Angabe von Zeilenbereichen.

Syntax

zeilenbereichsliste:: = { zeilenbereich }

zeilenbereich:: { zeilennummer | untere-grenze TO obere-grenze }

untere-grenze:: = { zeilennummer | FIRST }

obere-grenze:: = { zeilennummer | LAST }

Ein Zeilenbereich, der nur aus einer Zeilennummer besteht, bezieht sich auf die Zeile mit dieser Zeilennummer, falls eine solche Zeile existiert. Ansonsten ist er leer. Ein Zeilenbereich mit unterer und oberer Grenze umfaßt alle Zeilen mit Nummern, die größer oder

gleich der unteren und kleiner oder gleich der oberen Grenze sind. FIRST bezeichnet jeweils die erste Zeile des aktuellen Programms und LAST dessen letzte Zeile. Zeilennummern oder die Schlüsselworte FIRST und LAST müssen in einem Zeilenbereich stets so angegeben werden, daß die obere Grenze nicht kleiner als die untere Grenze ist.

In den meisten Kommandos können mehrere Zeilenbereiche verarbeitet werden, die in einer Zeilenbereichsliste angegeben werden. Dabei ist zu beachten, daß die obere Grenze eines Zeilenbereichs stets kleiner als die untere Grenze des in der Zeilenbereichsliste folgenden Zeilenbereichs sein muß. In einigen Kommandos darf die Zeilenbereichsliste fehlen. Dann bezieht sich das Kommando auf das ganze Programm, d. h., es wird der Zeilenbereich FIRST TO LAST bearbeitet.

Dateispezifikationen treten in den Kommandos auf, die BASIC-Zeilen von Magnetplatten oder Geräten lesen bzw. ausgeben, sowie in den Kommandos ERRORLIST und PROTOCOL. In der Dateispezifikation kann ein Gerät, ein Pfad, ein Dateiname und ein Dateityp angegeben werden.

Syntax

dateispezifikation:: = gerät
(gerät:) (pfad) dateiname (.dateityp)

Der zur Bildung der Geräteangabe, der Pfadangabe, des Dateinamens und des Dateityps verwendbare Zeichensatz sowie die mögliche Länge dieser Angaben sind der Dokumentation des Betriebssystems zu entnehmen. Durch das BAS I C-Programmiersystem werden Dateispezifikationen nicht überprüft. Fehler werden nur dann bemerkt, wenn das Betriebssystem eine Dateispezifikation abweist. Fehlende Teile in der Dateispezifikation werden durch das BASIC-Programmiersystem ergänzt. Die entsprechenden Standardannahmen sind bei den einzelnen Kommandos abgegeben.

In den Beispielen bezeichnen die Geräteangaben A:, B:, C:... jeweils Magnetplatten, E: die virtuelle Magnetplatte, CON: das Terminal und PRN: den Drucker.

3.3. Ausgabe des Programmtextes auf das Bedienerterminal (LIST)

Mit dem LIST-Kommando werden der Programmtext oder Abschnitte davon auf dem Bedienterminal ausgegeben.

Syntax

list-kommando:: = **LIST** (zeilenbereichsliste)

Die auszugebenden Abschnitte werden durch die Zeilenbereichsliste festgelegt. Fehlt die Zeilenbereichsliste, wird der gesamte Programmtext ausgegeben. Die Ausgabe auf das Bedienterminal kann durch Eingabe von CTRUS angehalten und durch erneute Eingabe von CTRUS fortgesetzt werden.

Beispiele

Kommando	Wirkung
LI	Ausgabe des gesamten Programmtextes über Terminal
LIST FIRST TO 200	Ausgabe des Programmtextes ab Programmbeginn bis einschließlich der Zeile 200
LIST 10 TO 90,300 TO LAST	Ausgabe aller Zeilen von 10 bis 0 und von 300 bis Programmende

3.4. Ausgabe der Programmlste (PROTOCOL)

Mit dem PROTOCOL-Kommando werden Programmlisten erzeugt, die den Programmtext und bei einer vorausgegangenen Übersetzung eventuell Fehlermeldungen und Warnungen enthalten. Die Listenausgabe erfolgt seitenweise. **Jede** Seite beginnt mit einer Kopfzeile. Diese Kopfzeile enthält den Identifikationscode (s. Abschn. 2.) sowie Datum und Uhrzeit.

Syntax

protocol-Kommando:: = PROTOCOL [dateispezifikation] [zeilenbereichsliste]

Durch Angabe einer Zeilenbereichsliste können Programmabschnitte für die Listenausgabe ausgewählt werden. Fehlt die Zeilenbereichsliste, wird das ganze Programm ausgelistet. Die Listenausgabe erfolgt auf die in der Dateispezifikation angegebenen Datei bzw. auf das angegebene Gerät. Fehlt die Dateispezifikation, erfolgt die Ausgabe auf das Bedienterminal (CON:). Fehlen Teile der Dateispezifikation, wird wie folgt ergänzt:

– Dateiname und Typ fehlen

  Wenn als Gerät CON: bzw. PRN : angegeben ist, erfolgt die Ausgabe auf Bedienterminal bzw. auf Drucker. Bei anderen Gerätenamen wird als Dateiname der Identifikationscode (s. Abschn. 2.) und als Dateityp .LST ergänzt.

– Dateiname angegeben

  Fehlt die Geräteangabe, erfolgt die Listenausgabe auf das aktuelle Gerät. Ein fehlender Dateityp wird zu .LST ergänzt.

  Pfadangabe fehlt

  Es wird das Standardverzeichnis verwendet.

– Wird nur die Zeilenbereichsliste aber keine Dateispezifikation angegeben, darf der Schrägstrich vor dem ersten Zeilenbereich nicht fehlen, sonst wird die Zeilenbereichsliste als Dateispezifikation gedeutet.

Beispiele

Kommando	Wirkung
PRO	Ausgabe der gesamten Programmliste über Bedienterminal
PPRN:	Ausgabe der Programmliste über Drucker
PA: /FIRST TO 10000	Ausgabe der Programmliste ab Programmbeginn bis Zeile 10000 auf Gerät A: mit dem Identifikationscode als Dateiname und Dateityp .LST
P TOUREN. DMP	Ausgabe der Programmliste auf das aktuelle Gerät unter dem Dateinamen TOUREN und dem Dateityp. DMP

3.5. Ausgabe der Fehlerliste (ERRORLIST)

Dieses Kommando arbeitet wie das PROTOCOL-KOMMANDO (s. Abschn. 3.4.), jedoch erscheinen nur die jenigen Zeilen in der Liste, in denen bei einer vorausgegangenen Übersetzung Fehler gefunden oder Warnungen ausgelöst wurden. Das ERRORLIST-Kommando kann auf solche Programme angewendet werden, bei deren Übersetzung mindestens ein Fehler festgestellt wurde. Enthält ein Programm lediglich Warnungen, sind diese mit dem PPOTOCOL-Kommando auszugeben.

Syntax

errorlist-kommando:: = ERRORLIST (dateispezifikation)

Durch das Kommando werden alle fehlerhaften Zeilen des Programms ausgelistet. Die Listenausgabe erfolgt in die spezifizierte Datei. Fehlende Teile der Dateispezifikation werden wie beim PROTOCOL-Kommando ergänzt.

Beispiel

Kommando	Wirkung
ER	Ausgabe der Fehlerliste über das Bedienterminal
ER PRN:	Ausgabe der Fehlerliste auf den Drucker

3.6. Eingabe eines Programms von einer Datei (OLD)

Mit dem OLD-Kommando wird ein Programm von Magnetplatte in den BASIC-Arbeitsspeicher gelesen. Vorher wird der BASIC-Arbeitsspeicher in den Initialzustand gebracht. Die darin stehenden Informationen gehen dabei verloren.

Syntax

old-kommando:: = OLD [dateispezifikation]

Die Dateispezifikation bezeichnet das einzulesende Programm. Zu lesende Dateien können mit einem Texteditor oder mit dem SAVE-Kommando (s. Abschn. 3.8.) erstellt werden. Enthält die gelesene Datei Zeilen, die nicht mit einer Zeilennummer beginnen, erscheint die Meldung:

NO LINE, INSERTION NOT POSSIBLE

und die betreffende Zeile wird nicht eingefügt.

Liegt eine Zeilennummer ausserhalb des Bereiches 1 bis 50000 oder hat sie mehr als 5 Ziffern, wird eine neue Zeilennummer angefordert (s. Abschn. 3.1.)

Fehlende Teile der Dateispezifikation werden wie folgt ergänzt:

- Fehlt die Geräteangabe, wird das aktuelle Gerät zur Eingabe verwendet.
- Fehlt der Dateiname, wird er durch den Identifikationscode ersetzt (s. Abschn. 2.).
- Fehlt der Dateityp, wird .BAS angenommen.
- Fehlt die Pfadangabe, wird das Standardverzeichnis verwendet.

Falls im OLD-Kommando keine Dateispezifikation abgegeben ist, werden alle Angaben wie beschrieben ergänzt.

Beispiel

Der Identifikationscode sei BEST, das aktuelle Gerät F:

Kommando	eingelesene Datei
OLD A: STR.BAS	A: STR.BAS
O VXCOP	F: VXCOP.BAS
OLD B:	**B: BEST.BAS**
O	**F: BEST.BAS**

3.7. Hinzufügen von Programmzeilen aus einer Datei (ADD)

Mit dem ADD-Kommando werden BASIC-Zeilen gelesen und entsprechend ihrer Zeilennummer in das im BASIC-Arbeitsspeicher stehende aktuelle Programm eingefügt. Steht im BASIC-Arbeitsspeicher ein abarbeitbares Zwischenprogramm, so wird es gelöscht.

Syntax

add-kommando:: ADD [datei spezifikation]

Das ADD-Kommando wird in der Regel verwendet, um an ein BASIC-Programm häufig benötigte Prozeduren anzufügen. Dazu müssen die Zeilennummern geeignet gewählt werden.

Reserviert man für das Hauptprogramm und dessen Prozeduren und für jede häufig benötigte Prozedur bestimmte Zeilenbereiche, so können „Bibliotheken" gebildet werden, die mit dem ADD-Kommando an das aktuelle Programm angefügt werden, und aus denen die benötigten Prozeduren anschließend mit dem EXTRACT-Kommando (s. Abschn. 3.11.) ausgewählt werden.

Fehlende Teile in der Dateispezifikation werden wie beim OLD-Kommando (s. Abschn. 3. 6.) ergänzt.

Beispiele

Kommando	Wirkung
ADD B: MENUE.BAS	Fügt an das aktuelle Programm MENUE.BAS vom Gerät B: an.
AD GRAPH	Fügt an das aktuelle Programm GRAPH.BAS an. GRAPH.BAS wird vom aktuellen Gerät gelesen.

3.8. Ausgabe des aktuellen Programms in eine Datei (save)

Mit dem SAVE-Kommando werden der Text des aktuellen Programms oder Textabschnitte als Datei ausgegeben und damit für die spätere Wiederverwendung gesichert.

Syntax

save-kommando:: = SAVE [dateispezifikation] [/zeilenberereichslistej

Die Datei, in die der Programmtext geschrieben wird, wird durch die Dateispezifikation bestimmt. Ist unter der angegebenen Dateispezifikation bereits eine Datei vorhanden, wird diese überschrieben. Für fehlende Teile der Dateispezifikation werden die selben Standardannahmen wie beim OLD-Kommando (s. Abschn. 3.6.) getroffen, d. h., als Gerät wird das aktuelle Gerät, als Dateiname der Identifikationscode und als Typ wird .BAS angenommen. Fehlt die Dateispezifikation ganz, werden alle Teile so ergänzt. Fehlt die Pfadangabe, wird das Standardverzeichnis verwendet.

Die Zeilenbereichsliste legt fest, welche Teile des Programms gesichert werden. Fehlt die Zeilenbereichsliste, wird das ganze Programm gesichert.

Die entstehenden Textdateien können mit dem OLD-Kommando wieder eingelesen werden oder mit einem Texteditor bearbeitet werden.

Das SAVE-Kommando sollte nach jeder Programmänderung und nach Eingabe längerer Programmabschnitte gegeben werden, um bei Störungen Programmverluste zu vermeiden.

Beispiele

Das aktuelle Gerät sei B:, der Identifikationscode ROT4

Kommando	Wirkung
SAVE	Ausgabe des aktuellen Programms als ROT4.BAS auf B:
SAE:	Ausgabe von ROT4.BAS auf E:
SA ROT5.BAS	Ausgabe als ROT5.BAS auf B:
SA PRN:	Druck des Programms ohne evtl. enthaltenen Fehlermeldungen und Warnungen
SA A:UP.BAS/10000	Sichern des Programmtextes ab Zeile 10000
TO LAST	bis Programmende als UP. BAS auf Gerät A:

3.9. Auswahl des aktuellen Gerätes (LOG)

Das aktuelle Gerät kann während der Arbeit mit einem BASIC-Programmiersystem geändert werden. Dazu verwendet man das LOG-Kommando.

Syntax

i^g-kommando:: = LOG gerät [:1

gerät:: = buchstabe

Sollten nacheinander mehrere Kommandos mit demselben Gerät ausgeführt werden (z. B. OLD-, SAVE- oder ADD-Kommandos) und dieses Gerät ist momentan nicht das aktuelle, so ist es sinnvoll, das zu benutzende Gerät festzulegen. Dann kann die explizite Geräteangabe bei diesen Kommandos entfallen.

Mit dem LOG-Kommando kann auch festgelegt werden, wo der BASIC-Arbeitsspeicher beim Verlassen des Systems abgelegt wird bzw. von welchem Gerät er bei Restart gelesen wird (s. Abschn. 2.).

Die durch das LOG-Kommando bewirkte Änderung des aktuellen Gerätes bleibt nach Verlassen des BASIC-Programmiersystems wirksam:

Beispiel

Kommando	Wirkung
LOG F:	Das aktuelle Gerät ist danach F
LOG A	Das aktuelle Gerät wird A

3.10. Löschen von Zeilen (DELETE)

Das DELETE-Kommando ermöglicht das Löschen von beliebigen Teilen des aktuellen Programms.

Syntax

delete-kommando:: = DELETE zeilenbereichsliste

Die in der Zeilenbereichsliste spezifizierten logischen Zeilen des aktuellen Programms werden aus dem Programmtext gelöscht.

Sind in der Zeilenbereichsliste einzelne Zeilen oder Zeilenbereiche angegeben, die nicht im Programm enthalten sind, werden die betreffenden Zeilen oder Zeilenbereiche ohne Wirkung übergangen.

Soll nur eine einzelne Zeile gelöscht werden, kann das auch erfolgen, indem nur die Zeilennummer, gefolgt vom Zeilenendezeichen, angegeben wird.

So haben z. B. die beiden Eingabefolgen:

DEL 5,17,39
und
5 (EOL)
17 (EOL)
39 (EOL) die gleiche Wirkung.

Beispiele

Kommando	Wirkung
DELETE 900 TO LAST	Löschen aller Zeilen ab Zeile 900 bis Programmende
D FIRST TO LAST	Löschen des gesamten Programms
D5,7,15,17	Löschen der Zeilen 5,7,9,15 und 17

3.11. Extrahieren von Zeilen (EXTRACT)

Mit dem EXTRACT-Kommando schneidet man aus dem aktuellen Programm die angegebenen Zeilen heraus, während die übrigen Zeilen gelöscht werden.

Syntax

extract-kommando:: = EXTRACT zeilenbereichsliste

Das EXTRACT-Kommando stellt damit eine Umkehrung des DELETE-Kommandos dar. Während im DELETE-Kommando in der Zeilenbereichsliste zu löschende Abschnitte des Programmtextes angegeben werden, gibt die Zeilenbereichsliste im EXTRACT-Kommando die Abschnitte an, die übrigbleiben sollen. Das Kommando kann vorteilhaft zum Herausschneiden wiederverwendbarer Prozeduren aus BASIC-Programmen angewendet werden.

Beispiele

Kommando	Wirkung
OLD A: MATRIX	Einlesen von A: MATRIX. BAS Diese Datei enthält die benötigte Prozeduren
EX 6300 TO 6545,6700 TO LAST	Herausschneiden der benötigten Teile
LOG C:	Auswahl von C: als aktuelles Gerät
ADD MAIN	Anfügen des Hauptprogramms C: MAIN.BAS
SAV MAIN	Sichern des kompletten Hauptprogramms

3.12. Neunumerieren von Zeilen (RENUMBER)

Mit dem RENUMBER-Kommando können Zeilen neu numeriert werden. Das Programm muß übersetzt worden.sein und darf keine Fehler enthalten.

Syntax

renumber-kommando:: =

RENUMBER [zeilenbereich] [ATzeilennummer [STEP schrittweite]] | [STEP schrittweite [ATzeilennummer]]

schrittweite:: = (ziffer)5

Ist im Kommando ein Zeilenbereich angegeben, so werden die Zeilen in diesem Zeilenbereich neu numeriert. Fehlt der Zeilenbereich, wird das ganze Programm neu numeriert. Ist das Schlüsselwort AT angegeben, erhält die erste neu zu numerierende Zeile die nach AT angegebene Zeilennummer. Die folgenden Zeilen des neu zu numerierenden Bereichs erhalten jeweils die um die Schrittweite erhöhte Nummer der Vorgängerzeile. Die Schrittweite kann nach dem Schlüsselwort STEP explizit angegeben werden. Fehlt die AT-Angabe, wird AT 100 angenommen. Fehlt die STEP-Angabe, gilt STEP 10.

Die erste Zeile des neu zu numerierenden Bereichs darf keine niedrigere Zeilennummer erhalten, als eine evtl. davorstehende Zeile, **die letzte Zeile dieses Bereiches darf keine höhere** Zeilennummer erhalten als die evtl. darauffolgende Zeile. Der für Zeilennummern zulässige Bereich von 1 bis 50000 darf nicht verlassen werden. Wird eine dieser Bedingungen verletzt, erscheint eine Fehlermeldung und das Kommando wird nicht ausgeführt.

Mit dem Neunumerieren der Zeilen werden auch die Verweise auf diese Zeile in GOTO, GOSUB, IF MISSING- u.a. Anweisungen entsprechend korrigiert. Damit diese Verweise vollständig aufgefunden werden, muß das Programm vorher übersetzt sein und darf keine Fehler enthalten.

Beim Numerieren geht das vorhandene abarbeitbare Zwischenprogramm verloren, d. h. es muß neu übersetzt werden.

Beispiele

Kommando	Wirkung
REN	**Neunumerieren des gesamten Programms in Zehnerschritten. Die erste Zeile enthält die Zeilennummer** 100
REN 9000 TO **12000** AT 10000 STEP 5	**Die Zeilen von 9000 bis** 12000 **werden in Fünferschritten neu numeriert. Die Zeile 9000 erhält die neue Zeilennummer** 10000.

3.13. Laden und Abarbeiten des aktuellen Programms (RUN)

Das RUN-Kommando bewirkt die Abarbeitung des aktuellen Programms.

Syntax

run-kommando:: = RUN

Existiert bereits ein abarbeitbares Zwischenprogramm, wird dieses aus dem BASIC-Arbeitsspeicher in den Hauptspeicher geladen und dort abgearbeitet. Existiert zum aktuellen Programm kein Zwischenprogramm, wird das aktuelle Programm vorher automatisch übersetzt und auf diese Weise das abarbeitbare Zwischenprogramm hergestellt. Treten bei der Übersetzung Fehler auf, erscheint eine Mitteilung auf dem Bedienterminal

ERROS XX (XX = Zahl der Fehler)

und das Programm gelangt nicht zur Ausführung. Treten bei der Übersetzung Warnungen auf, wird das durch die Mitteilung

WARNINGS XX (XX = **Zahl** der Warnungen)

angezeigt. Das Programm wird trotzdem ausgeführt.

Ein vorhandenes abarbeitbares Zwischenprogramm geht verloren, wenn Zeilen eingefügt bzw. gelöscht werden (OLD-, ADD-, DELETE- oder EXTRACT-Kommando) oder wenn Zeilen neu numeriert werden (RENUMBER-Kommando).

3.14. Wiederholbare Abarbeitung des aktuellen Programms (START)

Soll ein Programm mehrmals nacheinander ausgeführt werden, kann nach dem ersten RUN-Kommando das Laden des abarbeitbaren Zwischenprogramm vom BASIC-Arbeitsspeicher eingespart werden, in dem man das im Hauptspeicher stehende Programm mit dem START-Kommando abarbeitet.

Syntax

start-kommando:: = START

Ging dem START-Kommando kein **RUN-Kommando** unmittelbar voraus, wird anstelle des START-Kommandos ein RUN-Kommando ausgeführt.

3.15. Übersetzung des Programms (TRANSLATE)

Das TRANSLATE-Programm verwendet man, wenn ein abarbeitbares Zwischenprogramm erzeugt, aber nicht ausgeführt werden soll.

Syntax

translate-kommando:: = TRANSLATE

Das TRANSLATE-Kommando wird verwendet, wenn z. B. beim erstmaligen Übersetzen eines neuen Programms Fehler zu erwarten sind oder wenn abarbeitbare Zwischenprogramme erzeugt werden sollen, die durch andere BASIC-Programme mit der chain-Anweisung gestartet werden (s. Abschn. 4.).

4. Programmverkettung mit der CHAIN-Anweisung

Durch die CHAIN-Anweisung (siehe Sprachbeschreibung, Abschnitt 9.3.) können mehrere BASIC-Programme so miteinander verbunden werden, daß sie nacheinander ohne Bedienereingriffe ablaufen. Das zu startende Programm wird in der chain-Anweisung im Programmbeschreiber bezeichnet.

Vom zu startenden Programm müssen das abarbeitbare Zwischenprogramm, der Programmtext und in speziellen Fällen auch Informationen aus den Überßetzertabellen vorhanden sein. Diese Informationen stehen nur im BASIC-Arbeitsspeicher zur Verfügung. Deshalb muß für das zu startende Programm zunächst ein Abzug des BASIC-Arbeitsspeichers hergestellt **werden,** nachdem das betreffende BASIC-Programm fehlerfrei übersetzt wurde. Dazu können die Kommandos TRANSLATE und RESTART verwendet werden (s. Abschn. 3.16. und 2.). Der Programmschreiber in der CHAIN-Anweisung muß den Namen (Identifikationscode) des gesicherten BASIC-Arbeitsspeichers mit dem zu startenden Programm enthalten. Eine Geräte- oder Typenangabe ist im Programmbeschreiber nicht zulässig. Als Gerät wird stets das aktuelle Gerät und als Typ stets .DAT ergänzt. Als Pfadangabe wird das Standardverzeichnis verwendet.

Bei der Ausführung der CHAIN-Anweisung wird der durch den Programmschreiber bestimmte Abzug des BASIC-Arbeitsspeichers vom aktuellen Gerät in den BASIC-Arbeitsspeicher eingelesen, und das darin vorhandene abarbeitbare Zwischenprogramm wird sofort abgearbeitet.

Wird unter dem betreffenden Programmschreiber keine Datei vom Typ .DAT auf dem aktuellen Gerät gefunden, bleibt das aktuelle Programm erhalten und es wird eine nicht triviale Ausnahme ausgelöst. Ist eine solche Datei vorhanden, die jedoch keinen BASIC-Arbeitsspeicher oder einen BASIC-Arbeitsspeicher ohne abarbeitbares Zwischenprogramm enthält, geht das aktuelle Programm verloren und die Abarbeitung wird abgebrochen.

Beispiel

Gegeben seien folgende Programme auf dem Gerät A:

```
10 PROGRAM P1
20 PRINT „P1 STARTET P2"
30 CHAIN „P2"
40 END
```

```
10 PROGRAM P2
20 PRINT „P2 STARTET P3"
30 CHAIN „P3"
40 END
```

```
10 PROGRAM P3
20 PRINT „ENDE P3"
30 END
```

Die folgenden Kommandofolgen zeigen, wie die Abarbeitung der verketteten Programme vorzubereiten ist. Der Inhalt des BASIC-Arbeitsspeichers soll jeweils auf dem Gerät B: abgelegt werden.

RES/DE	Restart des Programmsystems mit dem Indifikationscode P2
IDENTIFICATION: P2	
LOG B:	Auswahl von B: als aktuelles Gerät
OLD A:	Lesen von A: P2.BAS als aktuelles Programm
T	Übersetzung
RES	Restart, der Arbeitsspeicher wird als B: P2.DAT abgelegt
IDENTIFICATION: P3	Neuer Identifikationscode ist P3
OLD A:	Lesen von A: P3.BAS
T	Übersetzen
RES	Restart, der Arbeitsspeicher wird als B: P3.DAT abgelegt
IDENTIFIKATION: P1	
OLD A:	P1 einlesen
RUN	Übersetzen, Laden und Abarbeiten von P1

Nach Abarbeitung der drei verketteten Programme ist der Identifikationscode P3. Im BASIC-Arbeitsspeicher steht das Programm P3 (Programmtext und abarbeitbares Zwischenprogramm).

Enthält ein neu eingegebenes Programm eine CHAIN-Anweisung, muß der Programmtext vor der Abarbeitung dieses Programms gesichert werden (z. B. mit dem SAVE-Kommando), da er sonst verlorengeht, wenn die CHAIN-Anweisung abgearbeitet wird.

5. Systemmitteilungen

5.1. Mitteilungen bei der Programmerstellung

Dieser Abschnitt umfaßt alle Mitteilungen, die bei der Eingabe von BASIC-Zeilen oder bei der Ausführung der im Abschnitt 3. beschriebenen Kommandos (außer RUN, TRANSLATE und START) erscheinen können.

Zur Kennzeichnung von Fehlern werden Kommandos zum Teil bis zum fehlerverursachenden Parameter unterstrichen bzw. Fehlermeldung beginnt mit Sternzeichen. Diese Kennzeichnungen sind im folgenden **Text nicht ausgewiesen. Die Meldungen sind** alphabetisch geordnet.

DISK FULL

diese Mitteilung erscheint, **wenn die Ausgabe von Programmzeilen, Listen oder des** Arbeitsspeichers auf eine Datei **abgebrochen werden muß,** da auf **dem betreffenden** Datenträger kein Platz mehr zur Verfügung steht.

ILLEGAL AT **PARAMETER**

IM renumber-Kommando steht **nach dem Schlüsselwort AT keine oder eine unzulässige Zeilennummer.**

ILLEGAL CHAR(S) REPLACED BY SPACE(S)

Eine eingegebene Zeile enthält nicht druckbare Zeichen. Diese Zeichen werden durch Leerzeichen ersetzt.

ILLEGAL COMMAND, MISSING PARAMETERS

Diese Meldung erscheint, wenn in einem Kommando Parameter fehlen. (Z. B. wenn im delete- oder im extract-Kommando die Zeilenbereichsliste fehlt.

ILLEGAL LINE NUMBER

In einem Kommando oder in einer BASIC-Zeile wurde eine Zeilennummer angegeben, die nicht im Bereich von 1 bis 50000 liegt, oder die aus mehr als fünf Ziffern besteht. Ein entsprechendes Kommando wird abgelehnt. Tritt dieser Fehler bei Eingabe einer BASIC-Zeile auf, wird eine neue Zeilennummer angefordert (s. Abschn. 3. 1.).

ILLEGAL RENUMBER PARAMETERS

Syntaxfehler im renumber-Kommando

ILLEGAL STEP PARAMETERS

Im renumber-Kommando steht nach dem Schlüsselwort STEP keine Schrittweite bzw. es wurde dort eine unzulässige Schrittweite angegeben.

NO ERRORS DETECTED IN THIS PROGRAM

Es wurde ein errorlist-Kommando verwendet, obwohl das Programm keine Fehler enthält bzw. noch nicht übersetzt worden ist.

NO LINE, INSERTION NOT POSSIBLE

Es wurde eine Folgezeile eingegeben, der keine BASIC-Zeile unmittelbar vorausging.

PARAMETER LIST TOO LARGE

Es sind in einer Zeilenbereichsliste mehr Zeilenbereiche angegeben worden, als intern verarbeitet werden können.

RENUMBER NOT POSSIBLE – PROGRAM NOT SUCCPSFULL TRANSLATED

Das renumber-Kommando kann nicht ausgeführt werden, da das Programm noch nicht erfolgreich übersetzt wurde.

RENUMBERED LINES WOULD OVERLAY LINE NOT BEING RENUMBERED

Das renumber-Kommando kann nicht ausgeführt werden, da die letzte Zeile des neu zu numerierenden Bereichs eine höhere oder die gleiche Zeilennummer erhalten würde, als die erste Zeile nach diesem Bereich.

RENUMBERING WOULD CREATE UNDEFINED LINE NUMBERS

Das renumber-Kommando kann nicht ausgeführt werden, da Zeilennummern > 50000 entstehen würden.

SEGMENT ITEMS NOT IN INCREASING ORDER

Die Zeilenbereiche in einer Zeilenbereichsliste sind nicht aufsteigend notiert.

SEQUENTIAL 10 FAILURE
SYSTEM ERROR CODE: XX

Diese Meldung zeigt einen Fehler bei der Ein- bzw. Ausgabe von BASIC-Programmen mit dem Kommando OLD, ADD bzw. SAVE oder bei der Listenausgabe auf Dateien an. XX bezeichnet den vom Betriebssystem gelieferten Fehlercode. Mit diesem Fehlercode kann anhand der Betriebssystemdokumentation die Fehlerursache bestimmt werden.

TEXT MEMORY CONTAINS NO LINES
Es wurde ein Kommando gegeben, das sich auf Programmzeilen bezieht, ohne daß sich Programmzeilen im BASIC-Arbeitsspeicher befinden.

UNDEFINED SEGMENT ITEM
Diese Meldung bezeichnet einen syntaktischen Fehler in einer Zeilenbereichsliste.

5.2. Mitteilungen über Fehler bei der Arbeit mit dem BASIC-Arbeitsspeicher

FILE CONTAINS NO VIRTUAL STORAGE
Eine als BASIC-Arbeitsspeicher einzulesende Datei vom Typ DAT enthält keinen virtuellen Speicher oder die Datei wurde mit einer anderen Version des BASIC-Programmiersystems erzeugt.

OVERFOLW ERROR LIST
Diese Meldung erscheint, wenn eine an das Programm bei der Übersetzung angefügte Fehlerliste nicht mehr im BASIC-Arbeitsspeicher untergebracht werden kann.
Der Programmtext Ist vollständig erhalten, jedoch erscheinen nicht alle ausgewiesenen Fehler in der Liste.

WORK BUFFER OVERFLOW
Während der Übersetzung eines Programms oder beim Hinzufügen von Programmtext steht nicht mehr genügend Platz im BASIC-Arbeitsspeicher zur Verfügung. Der laufende OVR-Teil wird durch den Kommando-Prozessor abgebrochen. Tritt der Fehler bei langen Programmen auf, sollten diese in mehrere Programme zerlegt werden.

WORK BUFFER TOO SMALL
Es steht nicht genügend Hauptspeicher zur Verfügung, um den BASIC-Arbeitsspeicher zu errichten. Falls im Hauptspeicher eine virtuelle Magnetplatte eingerichtet ist, sollte diese verkleinert werden. Dieser Fehler führt zum Abbruch der Arbeit mit dem BASIC-Programmiersystem.

WORK **BUFFER TOO** SMALL FOR GIVEN VS-FILE
Der bestehende BASIC-Arbeitsspeicher ist zu klein, um einen auf Datei gesicherten BASIC-Arbeitsspeicherinhalt einlesen zu können. Dieser Fehler kann nach Veränderung der Systemkonfiguration, insbesondere bei Vergrößerung der virtuellen Magnetplatte auftreten, wenn das System gestartet, ein restart-Kommando oder eine chain-Anweisung ausgeführt wird.

5.3. Mitteilungen bei der Programmübersetzung

5.3.1. Mitteilungen bei der lexikalen Analyse

Dieser Abschnitt umfaßt Fehlermeldungen und Warnungen, die bei der Verarbeitung elementarer Sprachkonstruktion wie:
Zeichen, Identifikatoren, Konstanten und Folgezeilen auftreten.

IDENTIFIER TOO LONG

Ein Identifikator ist länger als 31 Zeichen. Die überschüssigen Zeichen werden ignoriert.

INVALID REAL CONSTANT

Syntaktischer Fehler in einer Gleitkommakonstante.

QUOTE IS COMPLETED **(Warnung)**

In einer string-Konstanten fehlt das schließende Anführungszeichen. Dieses **Anführungszeichen** wird am Zeilenende automatisch ergänzt.

NO STANDARD CONFORMING CHARACTER **(Warnung)**

Eine string-konstante enthält unzulässige Zeichen. Das betreffende Zeichen **wird** in die interne Form der string-Konstante übernommen. Die Übersetzung wird fortgesetzt. Das Programm ist jedoch auf anderen Standard-BASIC-Systemen u. U. **nicht lauffähig, da** die Verwendung des Zeichens nicht standardgerecht **ist.**

REAL CONSTANT OVERFLOW

Bei der Konvertierung einer Gleitkommazahl tritt Zahlenüberlauf ein.

5.3.2. Syntaktische Fehler

Lexikale, syntaktische und semantische Fehler (s. Abschn. 5.3.3.) werden am Ende der Übersetzung durch die Ausschritten

ERRORS XX
WARNINGS XX

signalisiert, wobei XX für die Zahl der erkannten Fehler bzw. Warnungen steht. Die Programmzeilen, bei denen Fehler erkannt werden, können mit dem errorlist- oder protocol-Kommando aufgesucht werden. Die Stelle, an der der Fehler innerhalb der Zeile bemerkt wurde, wird durch Unterstreichung mit Sternzeichen (Fehler) oder Pluszeichen (Warnungen) markiert. Die Unterstreichung umfaßt den Teil der **Zeile,** der bis zur Erkennung des Fehlers übersetzt wurde.
Bei syntaktischen Fehlern ist die markierte Abbruchstell stets auch die fehlerverursachende Stelle, bei semantischen Fehlern (s. Abschn. 5.3.3.) wird durch die Unterstreichung lediglich die Stelle markiert, an der der Fehler erkannt wurde. Die Fehlerursache kann entweder in der gleichen Zeile im unterstrichenen Teil oder in vorhergehenden Zeilen liegen.

Syntaktische Fehler werden stets durch eine Meldung angezeigt, die mit dem Wort „Missing:" beginnt. Danach werden die Symbole angezeigt, die an der Abbruchstelle erwartet wurden. Die Symbole werden durch Schrägstriche getrennt.

Es werden nur soviel Symbole ausgegeben, wie auf einer Zeile untergebracht werden können.

Bei der Auswertung der Syntaxfehler sind folgende Besonderheiten zu beachten:

– Bei Syntaxfehlern in Ausdrücken (numerische, logische oder string) werden die möglichen Fortsetzungssymbole wegen der großen Vielfalt nicht detailliert ausgewiesen. Es erscheint dafür in der **Liste** de erwarteten Symbole die Zeichenreihe [EXPRESSION]

– TAIL COMMENT erscheint immer dann, wenn wahlweise ein Kommentar bzw. das Zeilenende erwartet wird (siehe Sprachbeschreibung, Abschnitt 4.2.)

Beispiel

```
44 LET X = (A+4) (B-3)
**** ********** ***
MISSING: \<EXPRESSION>\TAIL COMMENT\
```

– Fehlt in einer Zeile das erste Schlüsselwort (z. B. LET), so erscheinen in der Liste der erwarteten Symbole die Schlüsselworte aller an dieser Stelle möglichen Anweisungen sowie die Endesymbole noch offener Blockanweisungen, soweit sie in einer Zeile untergebracht werden können.

Beispiel

```
44 X = (A+4) (B-3)

MISSING: \END\LET\GOTO\GOSUB\GO\PRINT\INPUT\LINE\OPEN\CLOSE\
```

– Tritt in einer sonst korrekten next-, loop-, end-select-, end-if-, end-when-, end-handler-, end-sub-, end-funktion-, use- oder case-Zeile ein Syntaxfehler ein, so sind Blockanweisungen fehlerhaft geschachtelt oder die Struktur kann nach vorangegangenen Syntaxfehlern nicht mehr korrekt analysiert werden.

Beispiel

```
220FORI=1TO10
230 IF K (I) = 0 THEN
240 NEXT I! IF-BLOCK AUF ZEILE 230 NICHT ABGESCHLOSSEN
*** **
MISSING: \LET\GOTO\GOSUB\GO\PRINT\INPUT\LINE\OPEN\CLOSE\SET\
```

– Fehlende Felddeklarationen oder fehlende formale Feldparameter, fehlende Funktions- oder Subroutinedeklarationen führen dazu, daß Indizierungen oder Parameterlisten nach den betreffenden Identifikatoren nicht akzeptiert werden.

Beispiel

```
10 DECLARE NUMERIC F B (10 TO 20)

MISSING: \(\*\, \TAIL COMMENT\
20 LET B=F(1)
** ****** **
MISSING: \<EXPfESSION>\TAIL COMMENT\
```

(Fehler auf Zeile 20 als Folgefehler der fehlerhaften Deklaration auf Zeile 10.)

Folgt nach einer Programmeinheit weder Kommentar noch eine externe Prozedur, wird die Übersetzung abgebrochen. Der Rest des Propamms wird übergangen. Dieser Fehler kann als Folgefehler auftreten, wenn Blockanweisungen nicht korrekt geschachtelt sind.

Beispiel

```
100IFJ=KTHEN
110 LET FKT= 0
120 END IF
125 END IF
**********
MISSING: \TAIL COMMENT\
130 PRINT „ENDE ITERATION"
***  *
MISSING: \END OF FILLE\TAIL COMMENT\REM\EXTERNAL\
±++++
COMPILED UNTIL HERE
```

5.3.3. Semantische Fehler

Wie semantische Fehler im Programmtext bezeichnet werden, wurde bereits im Abschnitt 5.3.2. behandelt. Es sei darauf verwiesen, daß viele semantische Fehler als Folgefehler nach vorhergehenden syntaktischen Fehlern auftreten können. In Zweifelsfällen ist es zweckmäßig, zuerst die syntaktischen Fehler zu korrigieren und eine Neuübersetzung zu veranlassen.

** TABLE OVERFLOW **

Überlauf übersetzerinterner Tabellen bei zu umfangreichen Programmen.

ALREADY DEFINED AS STANDARDFUNKTION OR INTEGER

Ein in einer integer-Spezifikation des Vorspiels angegebener Identifikator (siehe Sprachbeschreibung, Anlage 1) bezeichnet entweder eine Standardfunktion oder wurde mehrfach spezifiziert.

ARITHMETIC OPTION IS INCOMPATIBLE WITH PREVIOUS CALLS

Eine externe Funktion oder Subroutine wird aus anderen externen Prozeduren bzw. dem Hauptprogramm gerufen und die ARITHMETIC-OPTION der rufenden Routinen stimmen untereinander oder mit der ARITHMETIC-OPTION der gerufenen Routine nicht überein, und es sind numerische Parameter oder string-Felder zu übergeben bzw. es handelt sich um eine externe Funktion mit numerischem Ergebnis.

ARRAY TOO LARGE

Es wurde ein Feld deklariert, das mehr Elemente enthält, als adressierbar sind.

CHANNEL NUMBER NOT IN RANGE 1 ...999

Die Nummer eines formalen Kanals in der Parameterliste einer Subroutine liegt nicht im Bereich von 1 bis 99.

COMPILED UNTIL HERE

Kennzeichnung des Abbruchs der Übersetzung (s. Abschn. 5.3.2.).

CONTROL VARIABLE MUST BE SINGLE VARIABLE

Als Laufvariable in einer for-Anweisung wurde eine indizierte Variable angegeben.

DATA STORAGE EXHAUSTED
In einer Programmeinheit wurden mehr Daten (Variable, Felder, Zeilen in der internen Datenliste) vereinbart, als adressiert werden können.

DECLARED INTERNAL FUNKTION 'XXX' HAS LEFT UNDEFINED
Eine in einer declare-Anweisung deklarierte interne Funktion oder interne Subroutine XXX wurde bis zum Ende dieser Programmeinheit nicht definiert.

DETACHED HANDLER 'XXX' HAS LEFT UNDEFINED
Eine in einer Programmeinheit benutzte selbständige Ausnahmeroutine XXX wurde in dieser Programmeinheit nicht definiert.

ERROR: ILLEGAL JUMP AT LINE XXXXX TO LINE YYYYY
Die Zeile XXXXX enthält einen Sprung zur Zeile YYYYY. Dieser Sprung ist nicht zulässig (Einsprung in Blockanweisung, interne Prozedur, Sprung zur ersten Zeile einer Prozedur, Sprungziel außerhalb der Programmeinheit).

ERROR: ILLEGAL USING OF LINE NUMBER XXXXX
AS DATA OR IMAGE LABLE

- Im Programm existiert eine restore-Anweisung mit der Zeilennumer XXXXX nach dem Schlüsselwort RESTORE und die Zeile XXXXX existiert nicht oder ist keine Data-Anweisung.
- Im Programm existiert eine print-Anweisung mit einem Formatverweis und die durch den Formatverweis bezeichnete Zeile ist keine Image-Zeile oder sie existiert nicht.

EXIT DO – OUT OF DO BLOCK
Eine exit-do-Anweisung steht außerhalb einer do-Schleife.

EXIT FOR – OUT OF FOR BLOCK
Eine exit-for-Anweisung steht außerhalb einer for-Schleife.

EXIT FUNCTION – OUT OF FUNCTION
Eine exit-function-Anweisung steht außerhalb einer externen oder internen Funktion.

EXIT SUB – OUT OF SUBROUTINE
Eine exit-sub-Anweisung steht außerhalb einer Subroutine.

EXCPRESSION MODE MISMATCHES WITH FUNCTION NAME
Der Typ des Ausdrucks in einer def-Anweisung stimmt nicht mit dem Typ des geforderten Resultatwertes überein (siehe Sprachbeschreibung, Abschnitt 9.1.1.).

FORMAL CHANNEL ALREADY DEFINED
In der Parameterliste einer Subroutine wurde mehrfach der gleiche formale Kanal angegeben.

FUNCTION NAME MAY APPEAR ONLY IN DEF LET STATEMENT
An den Namen einer Funktion darf der Resultatwert nur in einer def-let-Anweisung zugewiesen werden (siehe Sprachbeschreibung, Abschnitt 9.1.3.)

FUNCTION RESULT HAS BEEN LEFT UNDEFINED
Eine Funktion enthält keine def-let-Anweisung zur Zuweisung des Resultatwertes.

HANDLER BLOCK MISMATCH

Eine selbständige Ausnahmeroutine wurde in einer externen Prozedur benutzt, aber außerhalb dieser Prozedur definiert bzw. außerhalb einer internen Prozedur benutzt und in einer internen Prozedur definiert.

HANDLER NAME ALREADY IN USE

Der Name für eine selbständige Ausnahmeroutine wurde bereits zur Bezeichnung eines anderen Objektes verwendet.

HANDLER STATEMENT – OUT OF ERROR HANDLER

Eine retry-, continue- oder exit-handler-Anweisung steht außerhalb einer Ausnahmeroutine.

IDENTIFIER ALREADY IN USE

Ein in einer Deklaration verwendeter Identifikator wurde bereits zur Bezeichnung eines anderen Objektes benutzt.

IDENTIFIER IS NO SUBPROGRAM NAME

Der in einer call-Anweisung verwendete Routineidentifikator bezeichnet keine Subroutine (Subroutine nicht deklariert oder interne Subroutine nicht vorher definiert).

ILLEGAL MERGING NUMERIC AND INTEGER VARIABLES OR ARRAYS

In einer let-Anweisung sind als Zielvariablen gleichzeitig numerische- und integer-Variablen verwendet worden (siehe Sprachbeschreibung, Anlage 1).

ILLEGAL MERGING NUMERIC AND STRING

In einem Ausdruck wurden numerische- und string-Daten vermischt.

ILLEGAL NESTING HANDLER

In einer selbständigen Ausnahmeroutine soll eine weitere selbständige Ausnahmeroutine vereinbart werden.

ILLEGAL NUMBER OF PARAMETERS

Eine Prozedur enthält mehr oder weniger formale Parameter, als vorher übersetzte Aufrufe dieser Prozedur aktuelle Parameter enthielten.

ILLEGAL OPERATION WITH STRINGS

Zur Verknüpfung von zwei string-Primaeren wurde ein anderer Operator als „&" verwendet.

ILLEGAL PARAMETER MODE

Der Typ eines Parameters bei Aufruf oder Definition einer Prozedur stimmt nicht mit dem für diesen Parameter vorher verwendeten Typ überein.

ILLEGAL PROTECT IN HANDLER

Es wurde versucht, einen protect-Block in einer selbständigen Ausnahmeroutine anzugeben.

ILLEGAL REFERENCE TO ROUTINE IDENTIFIER

In einem Ausdruck wurde der Name einer externen Prozedur oder der (implizite) Name des Hauptprogramms verwendet (Identifikationscode).

ILLEGAL USE OF ACTUAL ARRAY

In einem Ausdruck wurde der Name eines Feldes ohne Indizierung verwendet.

ILLEGAL USE OF PRINT CONTROL FUNCTION

IN einem numerischen Ausdruck taucht ein Tabulator (Drucktrenner TAB) auf (siehe Sprachbeschreibung, Abschnitt 10.3.3.)

ILLEGAL USE OF RELATIONAL EXPRESSION

Es wurde ein logischer Ausdruck außerhalb einer bedingten Anweisung verwendet.

INTEGER BOUND REQUIRED

Es wurde im Vorspiel die Grenzwertspezifikation für integer-Feldgrenzen angegeben (siehe Sprachbeschreibung, Anlage 1). Eine untere bzw. obere Grenze verläßt den Bereich von –32768 bis 32767.

INTERNAL DEF. FUNCTION MAY NOT HAVE CHANNEL PARAMETER

An eine interne Subroutine dürfen keine Kanäle übergeben werden.

LABEL OVERFLOW

Überlauf des übersetzerinternen Markengenerators.

LAST EXPRESSION SHALL BE NUMERIC

Links vom Ende der Unterstreichung wurde ein numerischer Ausdruck gefordert.

LAST EXPRESSION SHALL BE RELATIONAL EXPRESSION

Links vom Ende der Unterstreichung wurde ein logischer Ausdruck gefordert.

LAST EXPRESSION SHALL GIVE A STRING

Links von Ende der Unterstreichung wurde ein string-Ausdruck gefordert.

LAST IDENTIFIER MAY NOT BE TARGET OF ASSIGNMENT

Links vom Ende der Unterstreichung wurde ein Identifikator angegeben, der nicht Ziel einer Zuweisung sein kann (z. B. der Name einer Standardfunktion).

LAST IDENTIFIER MAY NOT BE USED IN EXPRESSIONS

Der links von der Unterstreichung stehende Identifikator kann nicht in einem Ausdruck auftreten (z. B. ein Routineidentifikator).

LAST PARAMETER HAS ILLEGAL DIMENSION

Ein aktueller Parameter in einem Funktionsruf hat eine andere Dimension als bei vorhergehenden Funktionsrufen derselben Funktion oder als in der Funktionsdefinition angegeben.

LAST PARAMETER HAS ILLEGAL MODE

Der aktuelle Parameter in einem Funktionsruf hat einen anderen Typ als bei vorhergehenden Funktionsrufen derselben Funktion oder als in der Funktionsdefinition angegeben.

LOWER BOUND > UPPER BOUND

Es wurde eine Felddeklaration angegeben, in der die untere (explizite oder implizite) Indexgrenze größer als die obere Indexgrenze ist.

MISSING PARAMETERS

In einem Funktionsruf oder in einer call-Anweisung wurden weniger Parameter angegeben als in vorherigen Rufen oder in der Definition.

MISSING SUBSCRIPTS)
Es wurde ein Feld mit weniger Indizes benutzt, als dieses Feld Dimensionen hat.

MORE SUBSCRIPTS THAN DIMENSIONS
Es wurden zur Bildung einer indizierten Variablen mehr Indizes angegeben, als das Feld Dimensionen hat.

MORE THAN ONE DESTINATION IN DEF-LET-STATEMENT
Eine def-let-Anweisung enthält mehr als eine Zielvariable.

MULTIPLE HANDLER DEKLARATION
Es wurden mehrere selbständige Ausnahmeroutinen gleichen Namens definiert.

NAME I$ PREDEFINED AS STANDARDFUNCTION
Der in einer option-integer-Spezifikation des Vorspiels stehende Identifikator ist der Name einer Standardfunktion.

NEXT WITHOUT FOR
Eine next-Zeile enthält eine andere Laufvariable, als die gemäß Schachtelung der Laufanweisung zugehörige for-Zeile.

NO PARAMETERS OR SUBSCRIPT ALLOWED
Eine parameterlose Routine wurde mit Parametern aufgerufen.

OPTION ALREADY DEFINED
Eine Option wurde mehrfach definiert.

OPTION ALREADY USED
ES wurde eine Option in einer option-Anweisung angegeben, nachdem bereits Anweisungen übersetzt wurden, die diese Option auswerten. (z. B. Angabe einer Arithmetic-Option nach dem ersten numerischen Ausdruck, nach der ersten Deklaration vom Typ NUMERIC; angle-Option nach Benutzung einer trigonometrischen Standardfunktion).

PARAMETER IDENTIFIER ALREADY IN USE
Eine Liste mit formalen Parametern enthält einen bestimmten Identifikator mehr als einmal.

PARAMETER SHOULD BE AN ARRAY
In einem Funktionsaufruf wurde als Parameter ein Feld gefordert, es wurde jedoch eine einfache oder indizierte Variable angegeben.

PROGRAM NAME DEFAULTED TO 'XXX' (Warnung)
Ein Hauptprogramm enthält keine program-Anweisung. Als Routinename des Hauptprogramms wird der Identifikationscode 'XXX' angenommen.

RESERVED FUNCTION OR IDENTIFIER
Es wurde versucht, eine Routine mit dem Namen einer parameterlosen Standardfunktion zu definieren.

ROUTINE DEFINITIO MISMATCHES WITH INVOCATION
Eine Routine wurde als Subroutine gerufen, aber als Funktion definiert bzw. umgekehrt oder sie wurde als externe (interne) Routine deklariert und als interne (externe) definiert.

ROUTINE IDENTIFIER ALREADY IN USE
Es wurde eine Routine definiert und als Routineidentifikator ein Name angegeben, der bereits ein anderes Objekt bezeichnet.

SAME CONTROL VARIABLE IN NESTED LOOPS
Ineinander geschachtelte Schleifen haben die gleiche Laufvariable. Dieser Fehler ist als Folgefehler nach Fehlern in for-Anweisungen typisch.

SKIP REST WITHOUT CHANNEL
Eine Eingabeanweisung über Terminal enthält eine SKIP-REST-Angabe.

STRING(S) TOO LONG
Eine in einer Deklaration explizit angegebene string-Länge ist größer als die maximale string-Länge.

SYNTAX TOO COMPLEX
Überlauf des Rekursionskellers im Syntaxanalysator nach zu tief verschachtelten Blockanweisungen oder zu langen Parameterlisten.

TOO MANY DIMENSIONS
Es wurde ein Feld mit mehr als drei Dimensionen deklariert oder in einer formalen Parameterliste angegeben.

TOO MANY PARAMETERS
Der Ruf einer Subroutine oder einer Funktion enthält mehr Parameter als vorherige Rufe oder die entsprechende Routinedefinition.

USE EXLINE/EXTYPE ONLY IN HANDLER
Die Standardfunktionen EXLINE oder EXTYPE wurden außerhalb einer Fehlerbehandlungsroutine verwendet.

VARIABLE IS ATTACHED IN A FOR STATEMENT (Warnung)
Eine Zählschleife enthält Anweisungen, die den Wert der Laufvariablen ändern. Dieser Fehler erscheint auch als Folgefehler bei „SAME CONTROL VARIABLE IN NESTED LOOPS".

WARNING: EXTERMINAL FUNKTION/SUB 'XXX' HAS BEEN LEFT UNDEFINED (Warnung;
Die externe Funktion/Subroutine 'XXX' wurde im Programm deklariert, aber nicht definiert. Das Programm entspricht nicht dem Standard und ist auf anderen standardgerechten BASIC-Implementierungen u. U. nicht abarbeitbar. Bei Aufruf der nichtdefinierten Funktion oder Subroutine wird eine nichttriviale Ausnahme ausgelöst.

5.4. Mitteilungen bet der Programmbearbeitung

In der Abarbeitungsphase können folgende Fehler auftreten:

** NO INTERMEDIATE PROGRAM AVAILABLE
Eine chain-Anweisung wurde ausgeführt, aber der gelesene Abzug des BASIC-Arbeitsspeichers enthält kein abarbeitbares Zwischenprogramm. Das Programm, das die chain-Anweisung enthält, ist bereits überschrieben (s. Abschn. 4.).

STACK OR POOL OVERFLOW AT LINE XXX

Bei Abarbeitung der Zeile XXX läuft der string-Pool oder der Stack über. (String-Pool und Stack werden im Speicher gegenläufig geführt. Der Stack enthält die Werte sämtlicher Variablen und Felder und numerischer Werte, der string-Pool enthält alle string-Werte und alle Kanäle.)

*** TABLE OVERFLOW: PROGRAM IS TOO COMPLEX

Bei der Überführung des Zwischenprogramms vom BASIC-Arbeitsspeicher in den Hauptspeicher läuft die Markentabelle über.

Treten während der Programmabarbeitung Ausnahmen auf, werden diese mit der Anschrift

EXCEPTION (** FATAL **) xxxxxx AT LINE yyyyy nicht triviale Ausnahme

bzw.

EXCEPTION (NON FATAL) xxxxxx AT LINE yyyyy triviale Ausnahme

mitgeteilt, wobei xxxxxx der Ausnahmecode und yyyyyy die Nummer der verursachenden Zeile ist. Bei nicht trivialen Ausnahmen wird die Programmabarbeitung abgebrochen. Bei trivialen Ausnahmen, wird die Abarbeitung nach der Ausschrift fortgesetzt, falls es sich nicht um die Ausnahme 10007 (BREAK-Anweisung) oder um eine Ausnahme bei der Terminaleingabe handelt. Bei diesen Ausnahmen kann mit weiteren Kommandos entschieden werden, wie die Programmabarbeitung fortgesetzt werden soll.

Diese Kommandos werden mit:

ENTER ACTION (STANDARD IS fehleraktion, FOR HELP TYPE H):

angefordert.

Nach dieser Anforderung kann eines der im folgenden beschriebenen Kommandos eingegeben werden. Wird an Stelle eines Kommandos nur das Zeilenendezeichen eingegeben, wird die für diese Ausnahme bestimmte Standardfehleraktion ausgeführt, die in der Anforderung nach „STANDARD IS" angegeben war. Wird „H" eingegeben, werden alle Kommandos über Bedienterminal angezeigt, die im konkreten Fall zulässig sind.

Kommandos nach Ausnahme 10007 oder fehlerhafter Terminaleingabe:

ABORT	ABORT bewirkt den Abbruch der Programmabarbeitung
HELP	Anzeige der möglichen Kommandos über Bedienterminal

Kommandos nach fehlerhafter Terminaleingabe:

(Bei bestimmten Ausnahmen sind nicht alle diese Kommandos zulässig!)

RETRY COMPLETE	Wiederholung der kompletten Eingabeanweisung
RETRY [CURRENT]	Wiederholung der Eingabeanweisung ab aktueller Eingabezeile
RETRY LAST	Wiederholung der Eingabeanweisung ab dem fehlerhaften Datenelement.
ADD	Anfügen eines Kommas an die aktuelle Eingabezeile und Fortsetzung der Programmabarbeitung
IGNORE	Ignorieren des Eingabefehlers und Fortsetzung der Programmabarbeitung

Kommandos nach Ausnahme 10007:

CONTINUE	Fortsetzung der Programmabarbeitung
DEBUG ON	Einschalten des Debug-Selektors
DEBUG OFF	Ausschalten des Debug-Selektors
TRACE ON [TO #zahl]	Einschalten des Tracens ggf. über angegebenen Kanal.
TRACE OFF	Ausschalten des Tracens

6. Unterbrechung des Programmlaufs

Bei Bedarf kann die Programmabarbeitung durch Eingabe von Control-C (gleichzeitiges Drücken der CTRL-Taste und C) zwangsweise unterbrochen werden.

Die Unterbrechung wird durch Auslösen der Ausnahme 10007 erreicht (s. Abschn. 5.4.). Die bei der Unterbrechung angezeigte Zeile ist noch nicht ausgeführt.

Werden während des Programmlaufs andere Zeilen als das Control-C eingegeben, ohne daß eine Eingabe angefordert wurde, gehen diese Zeichen verloren.

Druck: 2669111/14/8 Graphische Werkstätten Zittau-Görlitz, BT 2 Ag 706/82/87 30